TP1

Bases de programmation

Exercice 1

Écrire un programme demandant à l'utilisateur de rentrer une année et affichant une phrase indiquant si cette année est bissextile ou non.
On rappelle que les années multiples de 4 *sont bissextiles, mais les années multiples de* 100 *ne le sont pas sauf si elles sont multiples de* 400.
On rappelle également que ces règles datent de l'avènement du calendrier grégorien en 1582 *et qu'avant cette date, toutes les années multiples de* 4 *étaient bissextiles sans exception (calendrier julien).*

Exercice 2 (*Somme des premiers entiers*)

Écrire un programme qui demande un entier n à l'utilisateur, qui calcule la somme des carrés des n premiers entiers ($1^2+2^2+\cdots+n^2$) et qui affiche la phrase (ici pour $n=3$) :
"La somme des carrés des 3 premiers entiers vaut 14."

Exercice 3

1. Écrire un programme qui compte le nombre d'éléments d'une chaine de caractère définie au préalable.
2. Écrire un programme qui compte le nombre d'espaces dans d'une chaine de caractère définie au préalable.
3. Écrire un programme qui supprime les espaces dans une chaine de caractère et affiche cette nouvelle chaine.

Exercice 4 (*Rang du premier terme d'une suite dépassant un nombre donné*)

On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par : $\begin{cases} u_0 = 1 \\ u_{n+1} = 2u_n + 3 \quad \forall n \in \mathbb{N} \end{cases}$.

1. Écrire un programme qui calcule et affiche u_{100}.
2. Écrire un programme qui calcule et affiche tous les termes u_i pour i allant de 1 à 10.
3. Écrire un programme renvoyant le plus petit entier n tel que : $\sum_{k=0}^{n} u_k > 1000$.

Exercice 5

1. Écrire une fonction *max2* qui prend en paramètre deux réels et qui renvoie le maximum de ces deux réels.
2. En utilisant la fonction *max2*, écrire une fonction *max3* qui prend en paramètre trois réels et qui renvoie le maximum de ces trois réels.

Exercice 6

1. Écrire une fonction qui prend en paramètre un entier n et qui renvoie le booléen *True* si cet entier est divisible par la somme de ses chiffres et le booléen *False* sinon.
   On pourra transtyper cet entier en chaine de caractère pour accéder facilement à ses chiffres.
2. Écrire un programme qui renvoie le nombre d'entiers inférieurs à 1000 qui sont divisibles par la somme de leurs chiffres.

Exercice 7 (*Suite de Syracuse*)

On définit la suite de Syracuse par $u_0 \in \mathbb{N}^*$ et $u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair,} \\ 3u_n + 1 & \text{si } u_n \text{ est impair.} \end{cases}$.

La conjecture de Syracuse (non démontrée à ce jour) est la suivante :
« Quelle que soit la valeur initiale $u_0 \in \mathbb{N}^*$ de la suite de Syracuse, il existe un entier N tel que $u_N = 1$ ».

1. Écrire un programme qui demande à l'utilisateur de saisir le premier terme $u_0 \in \mathbb{N}^*$ et qui affiche tous les termes de la suite de Syracuse jusqu'à l'apparition du premier 1.
2. Modifier ce programme pour qu'il affiche le rang N de l'apparition du premier 1 ainsi que la plus grande valeur atteinte par la suite jusqu'à l'apparition du premier 1.

Exercice 8 (*Test de primalité*)

On rappelle qu'un entier n est premier si il possède exactement deux diviseurs : 1 et lui même.
On peut montrer que si un entier n ne possède aucun diviseur inférieur à $\sqrt{n}$, alors il est premier.

1. Écrire une fonction prenant en paramètre un entier n et renvoyant le booléen *True* si cet entier est premier et *False* sinon.
2. Écrire un programme renvoyant le nombre de nombres premiers inférieurs à 1000.
3. Écrire un programme renvoyant le plus petit nombre premier supérieur à 1001.

Exercice 9

Écrire un programme qui génère toutes les tables de multiplications de 1 à 10.

Exercice 10

Écrire trois programmes permettant de représenter chacune des trois figures ci-dessous.

```
****        *        ****
****        **       ***
****        ***      **
****        ****     *
```

Exercice 11

1. Écrire une fonction prenant en paramètre une liste d'entiers L et renvoyant la somme des éléments pairs de cette liste.
2. Écrire une fonction prenant en paramètre une liste de réels L et renvoyant la moyenne des éléments de cette liste.
3. Écrire une fonction prenant en paramètre une liste de réels L et renvoyant une autre liste ne contenant que les termes strictement négatifs de la liste L.
4. Écrire une fonction prenant en paramètre une liste de réels L et renvoyant une autre liste contenant les termes de L en sens inverse (du dernier au premier).
5. Écrire une fonction prenant en paramètres deux listes d'entiers L_1 et L_2 et renvoyant le booléen *True* si ces deux listes sont identiques et *False sinon*.

Exercice 12 (*Suite de Fibonacci*)

On considère la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ définie par $F_0 = 0$, $F_1 = 1$ et pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$.

1. Écrire un programme demandant à l'utilisateur un nombre entier n et affichant le n-ième terme de la suite de Fibonacci.
2. Écrire une fonction prenant en paramètre un entier n et renvoyant une liste contenant les n premiers termes de la suite de Fibonacci.

Exercice 13 (*Test de primalité*)

1. Écrire une fonction prenant en paramètre un entier n et renvoyant le nombre de diviseurs de cet entier.
2. Comment utiliser la fonction précédente pour tester la primalité d'un entier.
3. Écrire une fonction prenant en paramètre un entier n et renvoyant une liste dont le terme d'indice i contient le booléen *True* si i est premier et le booléen *False* sinon.

Exercice 14

1. Écrire une fonction prenant en paramètres un élément e et une liste L et renvoie le nombre de fois ou l'élément e apparait dans la liste L.
2. Écrire une fonction prenant en paramètre une liste d'entiers L et qui renvoie le maximum de cette liste.
3. Écrire une fonction prenant en paramètre une liste d'entiers L et qui renvoie le deuxième maximum de cette liste.